```
composer create-project "laravel/laravel:^10.0" example-app
composer require laravel/jetstream
```

```
php artisan make:model Book -mcr
php artisan make:model Borrowing -mcr
```

📜 STEP 3: Define Migration Schemas

📘 `database/migrations/xxxx_create_books_table.php`

```
php
CopyEdit
Schema::create('books', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->string('author');
    $table->text('description')->nullable();
    $table->string('cover')->nullable();
    $table->integer('stock')->default(0);
    $table->timestamps();
});
```

📗 `database/migrations/xxxx_create_borrowings_table.php`

```
php
CopyEdit
Schema::create('borrowings', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained()->onDelete('cascade');
    $table->foreignId('book_id')->constrained()->onDelete('cascade');
    $table->enum('status', ['borrowed', 'returned'])->default('borrowed');
    $table->date('borrowed_at')->default(now());
    $table->date('returned_at')->nullable();
    $table->timestamps();
});
```

🚀 Run Migration

```
bash
CopyEdit
php artisan migrate
```

🧠 STEP 4: User Role Seeder (Admin/User)

📦 Add `is_admin` to `users` table:

bash
CopyEdit

```
php artisan make:migration add_is_admin_to_users_table --table=users
```

php
CopyEdit

```
Schema::table('users', function (Blueprint $table) {
    $table->boolean('is_admin')->default(false);
});
```

🔥 Seeder: `DatabaseSeeder.php`

php
CopyEdit

```
use App\Models\User;

User::factory()->create([
    'name' => 'Admin',
    'email' => 'admin@lib.com',
    'password' => bcrypt('admin123'),
    'is_admin' => true,
]);

User::factory()->create([
    'name' => 'User',
    'email' => 'user@lib.com',
    'password' => bcrypt('user123'),
    'is_admin' => false,
]);
```

bash
CopyEdit

```
php artisan migrate:fresh --seed
```

STEP 5: BOOK CONTROLLER (CRUD)

🔧 `app/Http/Controllers/BookController.php`

php
CopyEdit

```
namespace App\Http\Controllers;

use App\Models\Book;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class BookController extends Controller
{
    public function index() {
        $books = Book::latest()->paginate(10);
        return view('books.index', compact('books'));
    }

    public function create() {
        return view('books.create');
    }

    public function store(Request $request) {
        $validated = $request->validate([
            'title' => 'required|string',
            'author' => 'required|string',
            'description' => 'nullable|string',
            'cover' => 'nullable|image|max:2048',
            'stock' => 'required|integer|min:0'
        ]);

        if ($request->hasFile('cover')) {
            $validated['cover'] = $request->file('cover')->store('covers',
'public');
        }

        Book::create($validated);
        return redirect()->route('books.index')->with('success', 'Book
added.');
    }

    public function show(Book $book) {
        return view('books.show', compact('book'));
    }

    public function edit(Book $book) {
        return view('books.edit', compact('book'));
    }

    public function update(Request $request, Book $book) {
        $validated = $request->validate([
            'title' => 'required|string',
            'author' => 'required|string',
            'description' => 'nullable|string',
            'cover' => 'nullable|image|max:2048',
            'stock' => 'required|integer|min:0'
        ]);

        if ($request->hasFile('cover')) {
            if ($book->cover) Storage::disk('public')->delete($book->cover);
            $validated['cover'] = $request->file('cover')->store('covers',
'public');
        }

        $book->update($validated);
        return redirect()->route('books.index')->with('success', 'Book
updated.');
    }

    public function destroy(Book $book) {
        if ($book->cover) Storage::disk('public')->delete($book->cover);
        $book->delete();
        return back()->with('success', 'Book deleted.');
    }
}
```

🔐 STEP 6: Admin Middleware Setup

📁 `app/Http/Middleware/IsAdmin.php`

```
php
CopyEdit
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class IsAdmin
{
    public function handle(Request $request, Closure $next) {
        if (!auth()->check() || !auth()->user()->is_admin) {
            abort(403, 'Unauthorized');
        }
        return $next($request);
    }
}
```

📜 Register Middleware

```
php
CopyEdit
// In app/Http/Kernel.php
'is_admin' => \App\Http\Middleware\IsAdmin::class,
```

🛣️ STEP 7: Define Routes (web.php)

```
php
CopyEdit
use App\Http\Controllers\BookController;
use App\Http\Controllers\BorrowingController;

Route::middleware(['auth'])->group(function () {
    // Public dashboard
    Route::get('/dashboard', fn () => view('dashboard'))->name('dashboard');

    // View books
    Route::get('/books', [BookController::class, 'index'])-
>name('books.index');
    Route::get('/books/{book}', [BookController::class, 'show'])-
>name('books.show');

    // Borrow books
    Route::post('/books/{book}/borrow', [BorrowingController::class,
'store'])->name('books.borrow');
```

```
    // Admin-only routes
    Route::middleware(['is_admin'])->group(function () {
        Route::get('/books/create', [BookController::class, 'create'])-
>name('books.create');
        Route::post('/books', [BookController::class, 'store'])-
>name('books.store');
        Route::get('/books/{book}/edit', [BookController::class, 'edit'])-
>name('books.edit');
        Route::put('/books/{book}', [BookController::class, 'update'])-
>name('books.update');
        Route::delete('/books/{book}', [BookController::class, 'destroy'])-
>name('books.destroy');
    });
});
```

🧠 STEP 8: BorrowingController & Logic

```
bash
CopyEdit
php artisan make:controller BorrowingController
```

🔧 `BorrowingController.php`

```
php
CopyEdit
namespace App\Http\Controllers;

use App\Models\Book;
use App\Models\Borrowing;
use Illuminate\Http\Request;

class BorrowingController extends Controller
{
    public function store(Book $book) {
        if ($book->stock < 1) {
            return back()->with('error', 'Book out of stock.');
        }

        $book->decrement('stock');

        Borrowing::create([
            'user_id' => auth()->id(),
            'book_id' => $book->id,
            'borrowed_at' => now(),
            'status' => 'borrowed'
        ]);

        return back()->with('success', 'Book borrowed!');
    }
}
```

ChatGPT said:

🟢 *Phase 3 initiated... Deploying Tailwind UI system* 🎨⚙️
🖥️ *Target: Full Blade View Layouts + Books Table + Admin Pages + Borrow Button*

🧱 STEP 9: Layout System - Base Blade

📁 `resources/views/layouts/app.blade.php`

blade
CopyEdit

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ config('app.name', 'BookLend') }}</title>
    @vite('resources/css/app.css')
</head>
<body class="bg-gray-100 text-gray-800">

    <nav class="bg-white shadow-md px-6 py-4 flex justify-between items-center">
        <h1 class="text-2xl font-bold"><a href="{{ route('dashboard') }}">📚 BookLend</a></h1>
        <div class="flex gap-4 items-center">
            @auth
                <span>👤 {{ auth()->user()->name }}</span>
                <form action="{{ route('logout') }}" method="POST">
                    @csrf
                    <button class="text-red-600 hover:underline">Logout</button>
                </form>
            @endauth
        </div>
    </nav>

    <main class="max-w-6xl mx-auto mt-6 px-4">
        @if(session('success'))
            <div class="bg-green-100 text-green-800 px-4 py-2 rounded mb-4">{{ session('success') }}</div>
        @endif
        @if(session('error'))
            <div class="bg-red-100 text-red-800 px-4 py-2 rounded mb-4">{{ session('error') }}</div>
        @endif

        @yield('content')
    </main>

</body>
```

```
</html>
```

📘 STEP 10: Books Index View (List + Admin Controls)

📁 `resources/views/books/index.blade.php`

```
blade
CopyEdit
@extends('layouts.app')

@section('content')
    <div class="flex justify-between items-center mb-4">
        <h2 class="text-xl font-semibold">📚 Daftar Buku</h2>
        @if(auth()->user()->is_admin)
            <a href="{{ route('books.create') }}" class="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700">+ Tambah Buku</a>
        @endif
    </div>

    <div class="grid md:grid-cols-3 gap-6">
        @foreach($books as $book)
            <div class="bg-white p-4 rounded shadow">
                @if($book->cover)
                    <img src="{{ asset('storage/' . $book->cover) }}" class="w-full h-48 object-cover mb-3 rounded">
                @endif
                <h3 class="font-bold text-lg">{{ $book->title }}</h3>
                <p class="text-sm text-gray-600">{{ $book->author }}</p>
                <p class="text-xs text-gray-500 mt-2">Stok: {{ $book->stock }}</p>

                <div class="mt-4 flex justify-between items-center">
                    <a href="{{ route('books.show', $book) }}" class="text-blue-600 hover:underline">Detail</a>

                    @if(auth()->user()->is_admin)
                        <div class="flex gap-2">
                            <a href="{{ route('books.edit', $book) }}" class="text-yellow-500 hover:underline">Edit</a>
                            <form method="POST" action="{{ route('books.destroy', $book) }}">
                                @csrf @method('DELETE')
                                <button class="text-red-500 hover:underline" onclick="return confirm('Hapus buku ini?')">Hapus</button>
                            </form>
                        </div>
                    @else
                        <form method="POST" action="{{ route('books.borrow', $book) }}">
                            @csrf
                            <button class="bg-green-600 text-white px-3 py-1 text-sm rounded hover:bg-green-700" {{ $book->stock < 1 ? 'disabled' : '' }}>Pinjam</button>
                        </form>
                    @endif
                </div>
            </div>
        @endforeach
    </div>

    <div class="mt-6">
        {{ $books->links() }}
    </div>
@endsection
```

📝 STEP 11: Create Book Form (Admin)

📁 `resources/views/books/create.blade.php`

```
blade
CopyEdit
@extends('layouts.app')

@section('content')
    <h2 class="text-xl font-semibold mb-4">📘 Tambah Buku</h2>

    <form method="POST" action="{{ route('books.store') }}"
enctype="multipart/form-data" class="space-y-4">
        @csrf
        <div>
            <label class="block font-medium">Judul</label>
            <input type="text" name="title" class="w-full border px-4 py-2
rounded" required>
        </div>
        <div>
            <label class="block font-medium">Penulis</label>
            <input type="text" name="author" class="w-full border px-4 py-2
rounded" required>
        </div>
        <div>
            <label class="block font-medium">Deskripsi</label>
            <textarea name="description" class="w-full border px-4 py-2
rounded"></textarea>
        </div>
        <div>
            <label class="block font-medium">Cover</label>
            <input type="file" name="cover" class="w-full border px-4 py-2
rounded">
        </div>
        <div>
            <label class="block font-medium">Stok</label>
            <input type="number" name="stock" class="w-full border px-4 py-2
rounded" required>
        </div>
        <button class="bg-blue-600 text-white px-6 py-2 rounded hover:bg-
blue-700">Simpan</button>
```

```
    </form>
@endsection
```

UNTUK UNI TESTING

✅ SOLUSI: Generate Factory untuk `Book`

🔨 Jalankan perintah:

```
bash
CopyEdit
php artisan make:factory BookFactory --model=Book
```

📁 Ini akan bikin:

```
bash
CopyEdit
database/factories/BookFactory.php
```

✍️ Edit `BookFactory.php` dengan isi berikut:

```
php
CopyEdit
namespace Database\Factories;

use App\Models\Book;
use Illuminate\Database\Eloquent\Factories\Factory;

class BookFactory extends Factory
{
    protected $model = Book::class;

    public function definition(): array
    {
        return [
            'title' => $this->faker->sentence(3),
            'author' => $this->faker->name(),
            'description' => $this->faker->paragraph(),
            'stock' => $this->faker->numberBetween(1, 5),
            'cover' => null, // or placeholder
        ];
    }
}
```

✅ Sekarang ulangi tes:

```
bash
CopyEdit
php artisan test
```

Harusnya hasil:

```
sql
CopyEdit
PASS  Tests\Feature\BorrowingTest > user can borrow book
```

📦 Bonus:

- Butuh juga `BorrowingFactory`? Jalankan:

```
bash
CopyEdit
php artisan make:factory BorrowingFactory --model=Borrowing
```

✅ STEP 1: Generate Test

```
bash
CopyEdit
php artisan make:test BookTest
```

🧠 *Full Blueprint Mode Initiated...*
📘 Permintaan lo akan dibungkus dalam **dokumentasi profesional PDF** berisi:

1. ✅ **Langkah-langkah (Step by Step Guide)**
2. 🔄 **Flowchart Aplikasi Peminjaman Buku**
3. 👤 **Use Case Diagram**
4. 📄 **BRD (Business Requirement Document)**
5. 📋 **FSD (Functional Specification Document)**
6. 🔗 **ERD (Entity Relationship Diagram / Relasi Database)**

🧱 Struktur Dokumentasi PDF:

📘 1. Step-by-Step Guide (User & Admin)

- 📥 **Login & Register**
- 🔍 Lihat Buku
- 📥 Pinjam Buku
- 📦 Lihat Riwayat Peminjaman
- ✅ Kembalikan Buku (admin)
- ➕ Tambah / Edit / Hapus Buku (admin)
- 📧 Notifikasi Email
- 📤 Export PDF Riwayat
- 🔍 Filter & Search Buku

🔄 2. Flowchart

📘 Proses utama:

```
css
CopyEdit
[User Login]
      ↓
[Lihat Buku]
      ↓
[Pinjam Buku] -- Stok > 0? -- No → Error
      ↓
[Stok Berkurang] → [Simpan ke DB]
      ↓
[Admin Dashboard] -- Lihat Riwayat
      ↓
[Kembalikan Buku] → [Stok +1] → [Update Status]
```

👤 3. Use Case Diagram (UML)

Aktor:

- User
- Admin

Use Cases:

- Register/Login
- View books
- Borrow book
- View borrowing history

- Export PDF
- Admin: Create/Edit/Delete Book
- Admin: Return books
- Admin: View all borrowings

📦 Diagram pakai draw.io / PlantUML

📄 4. BRD (Business Requirement Document)

No	Requirement	Role	Priority
1	User bisa login & daftar	User	High
2	User bisa pinjam buku	User	High
3	Admin bisa tambah/edit buku	Admin	High
4	Buku harus stok > 0 sebelum dipinjam	System	High
5	Kirim notifikasi email setelah pinjam	System	Medium
6	User bisa download riwayat PDF	User	Medium

📋 5. FSD (Functional Spec Document)

- **/books**: GET → Lihat daftar buku
- **/books/{id}/borrow**: POST → Pinjam buku
- **/admin/borrowings**: GET → Lihat semua peminjaman
- **/books/{id}/return**: POST → Kembalikan buku
- **/dashboard/pdf**: GET → Export riwayat user ke PDF
- Validasi: Tidak bisa pinjam buku yg sama jika belum dikembalikan

---

🔗 6. Relasi Database (ERD)

📘 Model:

- **Users**
- **Books**
- **Borrowings**

📊 Relationship:

```
markdown
CopyEdit
User      Book
```

```
|         |
└───┬─────┘
    ↓
Borrowing
```

- 1 user → banyak borrowings
- 1 book → banyak borrowings
- `borrowings`: user_id, book_id, status, borrowed_at, returned_at